МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

Тема: Линейные структуры данных: стек, очередь, дек

Вариант 9-в

Студент гр. 8304 _____ Барышев А.А.

Преподаватель _____ Фирсов К.В.

Санкт-Петербург

2019

Задание.

В заданном текстовом файле F записан текст, сбалансированный по круглым скобкам:

<текст>::=<пусто>|<элемент><текст>

<элемент>::=<символ>|(<текст>)

где <символ> – любой символ, кроме (,). Для каждой пары соответствующих открывающей и закрывающей скобок вывести номера их позиций в тексте, упорядочив пары в порядке возрастания номеров позиций:

а)закрывающих скобок; б)открывающих скобок.

Например, для текста A+(45-F(X)*(B-C)) надо напечатать:

а) 8 10; 12 16; 3 17; б) 3 17; 8 10; 12 16.

Цель работы.

Получить практические навыки по использованию стека, вспомнить, как работает стек, какую структуру имеет. Применить методы стека для решения поставленной задачи.

Описание программы.

Данная программа принимает строки из текстового файла до тех пор, пока не встретит в текстовом файле терминальное слово «END». Каждая строка помещается в функцию CloseBrackets, а затем в функцию OpenBrackets. Программа не ломается, работает корректно, если для определения, данного в задании, подобрать такие тесты, как пустая строка, две скобки, один символ или набор символов без скобок.

Функции.

Основные функции данной программы:

void CloseBrackets(std::string);

void OpenBrackets(std::string);

Рассмотрим функцию CloseBrackets. В данной функции создаётся стек на два элемента типа int. Далее последовательно проверяется каждый символ строки, начиная с первого, и если он оказывается соответствующим «)», то сохраняется позиция этого символа помещается на стек, и от неё начинается поиск символа «(». Если снова встретится символ «)», то будет увеличен счётчик, названный counterAfterBracket, который увеличивается с каждым новым таким символом «)», и уменьшается, как только встретит символ «(». Если при поиске при нахождении в строке символа «(» счётчик counterAfterBracket будет равен нулю, то counterOpenBrackets будет помещен на стек.

Функция OpenBrackets работает по аналогичному принципу: разница в том, что на стек помещается значение counter после того, как помещен номер позиции символа «)».

Тесты.

Данная программа тестировалась на «сложных» и «критических» наборах входных параметров, ошибки отсутствовали. Программа проверялась только на верных набор данных, при вводе строки с нечётным количеством скобок или неверно расставленным, программа *гарантированно уходит в бесконечный цикл*, но таких ошибок быть не может, так как тестовые данные всегда соответствуют определению. Вначале приведена строчка, а ниже соответствующий ей результат.

Результаты тестирования:

A+(45-F(X)*(B-C))

a) 8 10; 12 16; 3 17; b) 3 17; 8 10; 12 16;

fkefaw

a) b)

()

a) 1 2; b) 1 2;

ifslj(WD(DWDA(WADA(WDA))))

a) 19 23; 14 24; 9 25; 6 26; b) 6 26; 9 25; 14 24; 19 23;

(((((())))))

a) 6 7; 5 8; 4 9; 3 10; 2 11; 1 12; b) 1 12; 2 11; 3 10; 4 9; 5 8; 6 7;

(())

a) 39 40; 38 41; 37 42; 36 43; 35 44; 34 45; 33 46; 32 47; 31 48; 30 49; 29 50; 28 51; 27 52; 26 53; 25 54; 24 55; 23 56; 22 57; 21 58; 20 59; 19 60; 18 61; 17 62; 16 63; 15 64; 14 65; 13 66; 12 67; 11 68; 10 69; 9 70; 8 71; 7 72; 6 73; 5 74; 4 75; 3 76; 2 77; 1 78;

b) 1 78; 2 77; 3 76; 4 75; 5 74; 6 73; 7 72; 8 71; 9 70; 10 69; 11 68; 12 67; 13 66; 14 65; 15 64; 16 63; 17 62; 18 61; 19 60; 20 59; 21 58; 22 57; 23 56; 24 55; 25 54; 26 53; 27 52; 28 51; 29 50; 30 49; 31 48; 32 47; 33 46; 34 45; 35 44; 36 43; 37 42; 38 41; 39 40;

orkw3prw3krkwpk((DAWMDA)FSEFSF)

a) 17 24; 16 31; b) 16 31; 17 24;

fe(nwndsnjkna)

a) 3 14; b) 3 14;

(wndj(dw)wa)

a) 6 9; 1 12; b) 1 12; 6 9;

Выводы.

Были получены практические навыки по использованию стека, освежены знания по структуре и работе стека. На практике была решена задача с применением стека и его основных методов push и pop. Также был изучен деструктор на C++, который позволяет освобождать выделенную память.

КОД ПРОГРАММЫ.

```
#include <iostream>
#include <string>
#include <fstream>

std::ofstream fOut;
template <class Item>
class Stack {
    private:
      Item *s;
            int N, N1;
    public:
      Stack(int maxN) {
                  s = new Item[maxN];
                  N = 0;
                  N1 = maxN;
            }
      bool empty() const {
                  return N == 0;
            }
      void push(Item elem) {
                  s[N++] = elem;
                  if (N > N1) {
                        std::cerr << "Error!";
                  }
            }
      Item pop() {
            if(N != 0){
                        return s[--N];
                  }
                  else{
                        std::cerr << "Error!";
                        return 0;
                  }
            }
            ~Stack(){
                  delete [] s;
                  N1 = 0;
                  N = 0;
            }
};
```

```
void OpenBrackets(std::string textExpression){
	bool flag = true;
	int counter = 0;
	int counterAfterBracket = 0;
	int counterCloseBrackets = 0;
	Stack<int> stackPositions(textExpression.size());
	fOut << "	b)";
	while(textExpression[counter] != '\0'){
		if(textExpression[counter] == '('){
			counterCloseBrackets = counter + 1;
			while(flag){
				if(counterAfterBracket == 0 &&
textExpression[counterCloseBrackets] == ')'){
					stackPositions.push(counterCloseBrackets);
					stackPositions.push(counter);
					flag = false;
				}
				else if(textExpression[counterCloseBrackets] == '('){
					counterCloseBrackets++;
					counterAfterBracket++;
				}
				else if(textExpression[counterCloseBrackets] == ')'){
					counterCloseBrackets++;
					counterAfterBracket--;
				}
				else{
					counterCloseBrackets++;
				}
			}
			flag = true;
			counterCloseBrackets = 0;
			fOut << " " << stackPositions.pop() + 1 << " " <<
stackPositions.pop() + 1 << ";";
		}

		counter++;
	}
	fOut << std::endl;
}

void CloseBrackets(std::string textExpression){
	bool flag = true;
```

```
	int counter = 0;
	int counterAfterBracket = 0;
	int counterOpenBrackets = 0;
	Stack<int> stackPositions(2);
	fOut << textExpression << std::endl << "a)";
	while(textExpression[counter] != '\0'){
		if(textExpression[counter] == ')'){
			stackPositions.push(counter);
			counterOpenBrackets = counter - 1;
			while(flag){
				if(counterAfterBracket == 0 &&
textExpression[counterOpenBrackets] == '('){
					stackPositions.push(counterOpenBrackets);
					flag = false;
				}
				else if(textExpression[counterOpenBrackets] == ')'){
					counterOpenBrackets--;
					counterAfterBracket++;
				}
				else if(textExpression[counterOpenBrackets] == '('){
					counterOpenBrackets--;
					counterAfterBracket--;
				}
				else{
					counterOpenBrackets--;
				}
			}
			flag = true;
			counterOpenBrackets = 0;
			fOut << " " << stackPositions.pop() + 1 << " " <<
stackPositions.pop() + 1 << ";";
		}

		counter++;
	}
}

int main(){
	int count = 0;
	std::ifstream fEnter("TestData.txt");
	fOut.open("Result.txt");
	std::string textExpression;
```

```
	fEnter >> textExpression;
	while(textExpression != "END")
	{
		CloseBrackets(textExpression);
		OpenBrackets(textExpression);
		fEnter >> textExpression;
	}
	fEnter.close();
	fOut.close();
	return 0;
}
```